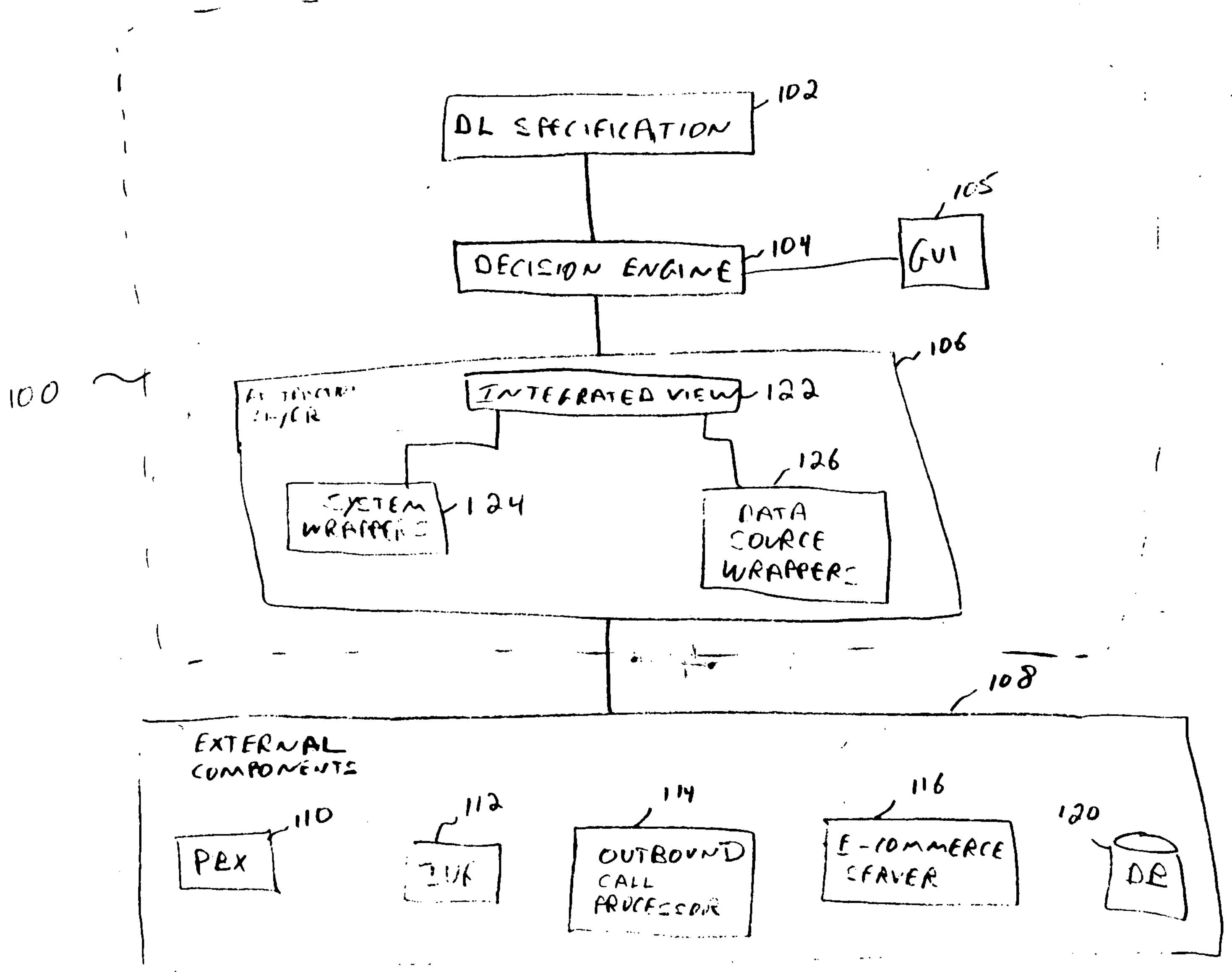


100
DL SPECIFICATION
102
DECISION ENGINE
104
GUI
105
106
INTEGRATED VIEW
122
SYSTEM WRAPPERS
124
DATA SOURCE WRAPPERS
126
108
EXTERNAL COMPONENTS
PBX
110
IVR
112
OUTBOUND CALL PROCESSOR
114
E-COMMERCE SERVER
116
DB
120

FIG 1

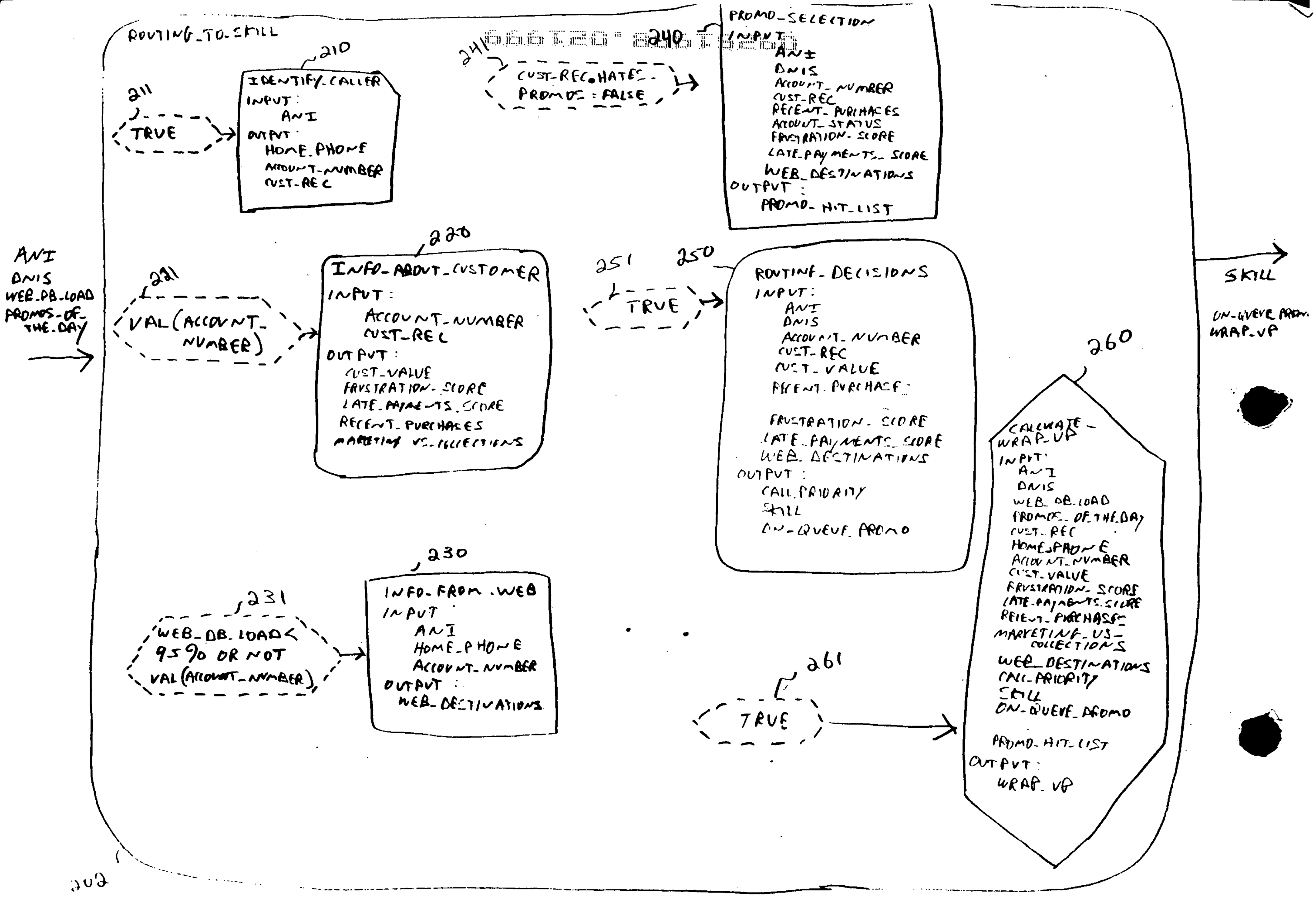

ROUTING_TO_SKILL
ANI
DNIS
WEB_DB_LOAD
PROMOS_OF_THE_DAY
211
TRUE
210
IDENTIFY_CALLER
INPUT:
ANI
OUTPUT:
HOME_PHONE
ACCOUNT_NUMBER
CUST_REC
221
VAL(ACCOUNT_NUMBER)
220
INFO_ABOUT_CUSTOMER
INPUT:
ACCOUNT_NUMBER
CUST_REC
OUTPUT:
CUST_VALUE
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE
RECENT_PURCHASES
MARKETING_VS_COLLECTIONS
231
WEB_DB_LOAD< 95% OR NOT VAL(ACCOUNT_NUMBER)
230
INFO_FROM_WEB
INPUT:
ANI
HOME_PHONE
ACCOUNT_NUMBER
OUTPUT:
WEB_DESTINATIONS
241
CUST_REC.HATES_PROMOS = FALSE
240
PROMO_SELECTION
INPUT:
ANI
DNIS
ACCOUNT_NUMBER
CUST_REC
RECENT_PURCHASES
ACCOUNT_STATUS
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE
WEB_DESTINATIONS
OUTPUT:
PROMO_HIT_LIST
251
TRUE
250
ROUTING_DECISIONS
INPUT:
ANI
DNIS
ACCOUNT_NUMBER
CUST_REC
CUST_VALUE
RECENT_PURCHASE
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE
WEB_DESTINATIONS
OUTPUT:
CALL_PRIORITY
SKILL
ON_QUEUE_PROMO
261
TRUE
260
CALCULATE_WRAP_UP
INPUT:
ANI
DNIS
WEB_DB_LOAD
PROMOS_OF_THE_DAY
CUST_REC
HOME_PHONE
ACCOUNT_NUMBER
CUST_VALUE
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE
RECENT_PURCHASES
MARKETING_VS_COLLECTIONS
WEB_DESTINATIONS
CALL_PRIORITY
SKILL
ON_QUEUE_PROMO
PROMO_HIT_LIST
OUTPUT:
WRAP_UP
SKILL
ON_QUEUE_PROMO
WRAP_UP
202

FIG. 2

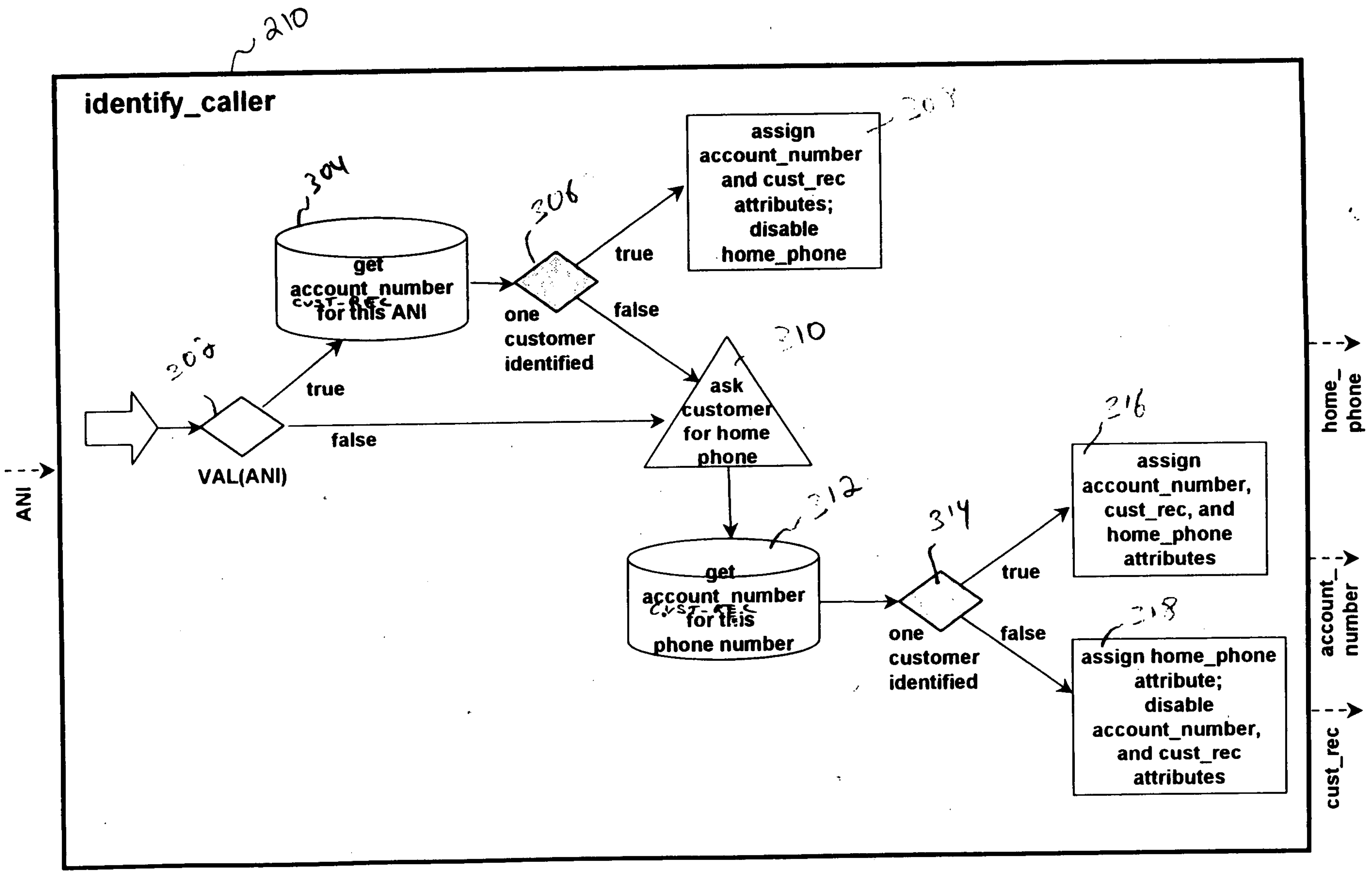
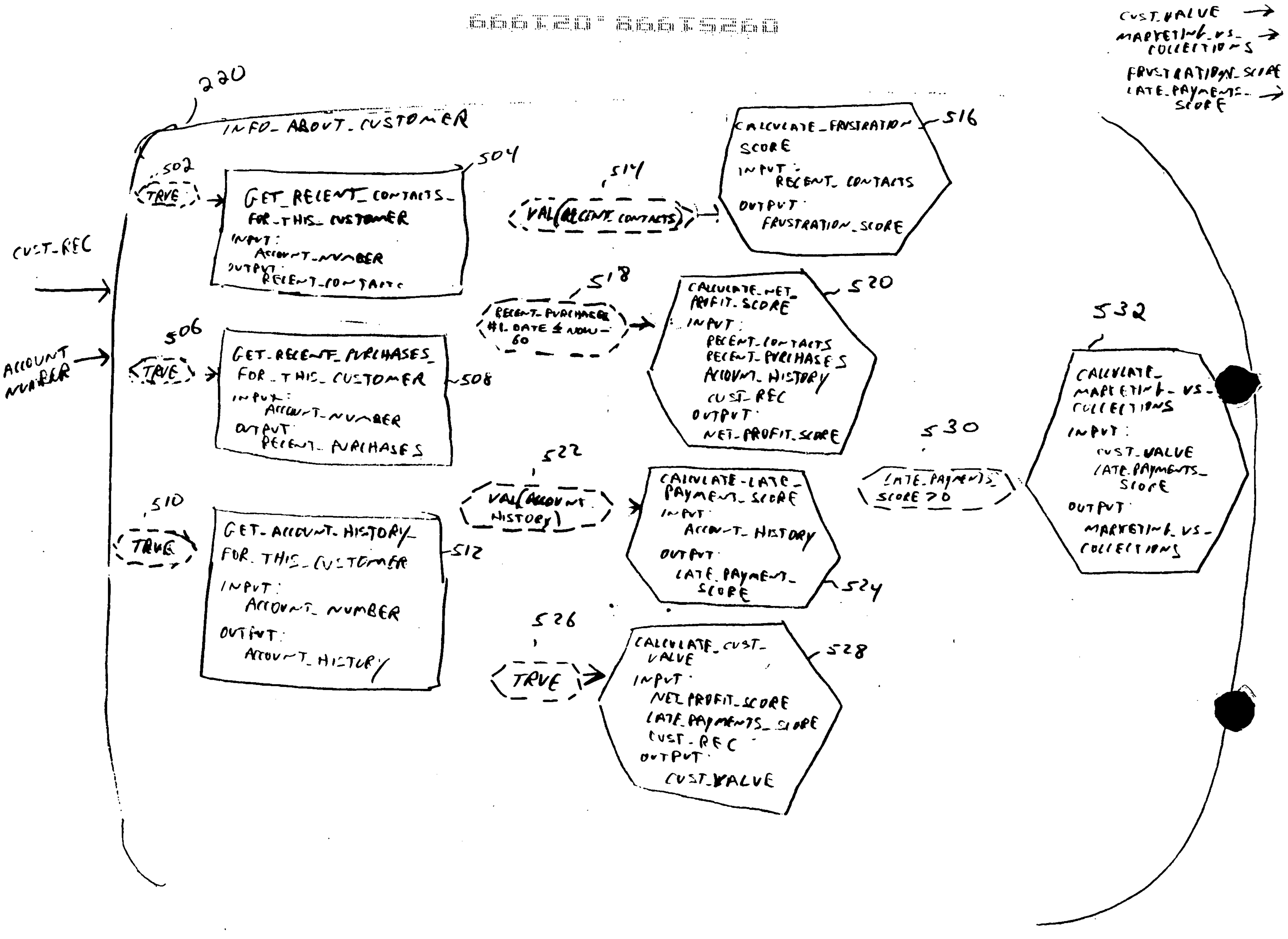

identify_caller
210
ANI
VAL(ANI)
208
true
false
304
get account_number cust-rec for this ANI
306
one customer identified
true
false
assign account_number and cust_rec attributes; disable home_phone
208
310
ask customer for home phone
312
get account_number cust-rec for this phone number
314
one customer identified
true
false
216
assign account_number, cust_rec, and home_phone attributes
218
assign home_phone attribute; disable account_number, and cust_rec attributes
home_phone
account_number
cust_rec

Fig. 3

```
Module:  identify_caller
   Submodule of:  routing_to_skill
   Input attributes:     ANI : 9digits
   Output attributes:    home_phone : 9digits
                         account_number : 15digits
                         cust_rec : tuple( name: string,
                                           address: string,
                                           card_color: {"platinum",
                                           "gold", "green"},
                                           hates_promos? : boolean,
                                           estimated_income_bracket :
                                           {"0-10K",">10K-20K",...,
                                           ">100K-150K", ">150K"},
                                           last_sent_bonus_check:date)
   Enabling condition:  true
   Type:           flowchart
   Computation:    See Fig. 3
   Side-effect:    yes
   Side Effect function: (IVR Dip)
```

FIG. 4



220
INFO_ABOUT_CUSTOMER
CUST-REC
ACCOUNT NUMBER
502
TRUE
GET_RECENT_CONTACTS_FOR_THIS_CUSTOMER
INPUT: ACCOUNT_NUMBER
OUTPUT: RECENT_CONTACTS
504
506
TRUE
GET_RECENT_PURCHASES_FOR_THIS_CUSTOMER
INPUT: ACCOUNT_NUMBER
OUTPUT: RECENT_PURCHASES
508
510
TRUE
GET_ACCOUNT_HISTORY_FOR_THIS_CUSTOMER
INPUT: ACCOUNT_NUMBER
OUTPUT: ACCOUNT_HISTORY
512
514
VAL(RECENT_CONTACTS)
CALCULATE_FRUSTRATION_SCORE
INPUT: RECENT_CONTACTS
OUTPUT: FRUSTRATION_SCORE
516
518
RECENT_PURCHASES #1.DATE ≤ NOW-60
CALCULATE_NET_PROFIT_SCORE
INPUT: RECENT_CONTACTS RECENT_PURCHASES ACCOUNT_HISTORY CUST_REC
OUTPUT: NET_PROFIT_SCORE
520
522
VAL(ACCOUNT_HISTORY)
CALCULATE_LATE_PAYMENT_SCORE
INPUT: ACCOUNT_HISTORY
OUTPUT: LATE_PAYMENT_SCORE
524
526
TRUE
CALCULATE_CUST_VALUE
INPUT: NET_PROFIT_SCORE LATE_PAYMENTS_SCORE CUST_REC
OUTPUT: CUST_VALUE
528
530
LATE_PAYMENTS_SCORE > 0
532
CALCULATE_MARKETING_VS_COLLECTIONS
INPUT: CUST_VALUE LATE_PAYMENTS_SCORE
OUTPUT: MARKETING_VS_COLLECTIONS
CUST_VALUE
MARKETING_VS_COLLECTIONS
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE

FIG. 5

```
Module:  info_about_customer
   Submodule of:  routing_to_skill
   Input attributes:    account_number
                        cust_rec

   Output attributes:   cust_value : [1..10]
                        frustration_score : [1..10]
                        late_payments_score : [1..10]
                        recent_purchases :list(tuple( date : date,
                                                      item : 20digit,
                                                      qty : int,
                                                      amount: $value ))
                        marketing_vs_collections : {"market",
                        "collect"}

Enabling condition:     VAL(account_number)
   Type:          declarative
   Side-effect:   no
```

Fig. 6

```
Module: info_from_web

   Submodule of:  routing_to_skill

   Input attributes:     ANI
                         home_phone
                         account_number

   Output attributes:    web_destinations : list(tuple(regions: set of
                                                  {"Australia","Asia",...
                                                   "NAmerica-US", "US"},
                                             itinerary:web_form_content,
                                             date_last_modified : date ))

   Enabling condition:   web_db_load < 95% or not VAL(account_number)

   Type:                 foreign

   Computation:          get_web_data(ANI, home_phone, account_number)

   Side-effect:          no
```

Fig. 7

```
Module:  promo_selection

    Submodule of:  routing_to_skill

    Input attributes:     ANI
                          DNIS
                          account_number
                          cust_rec
                          cust_value
                          recent_purchases
                          frustration_score
                          late_payments_score
                          web_destinations

    Output attributes:    promo_hit_list : list ( promo_message )

    Enabling condition:   cust_rec.hates_promos? = false

    Type:                 foreign

    Computation:          get_promo_hit_list(ANI, DNIS, account_number,
                          cust_rec, cust_value, recent_purchases,
                          account_status, frustration_score,
                          late_payments_score, web_destinations)
    Side-effect:          no
```

Fig. 8

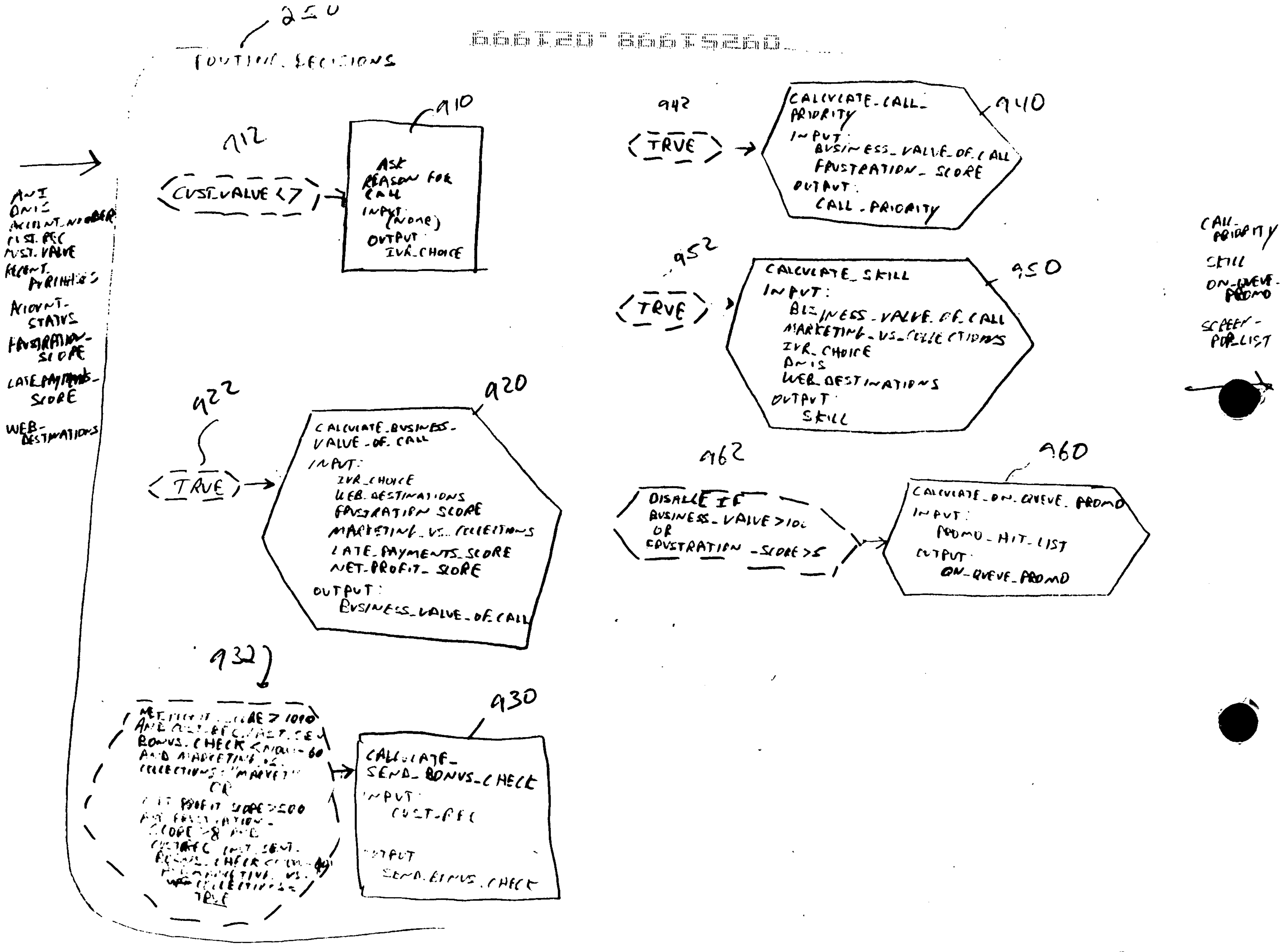

250
ROUTING DECISIONS
ANI
DNIS
ACCOUNT_NUMBER
CUST_REC
CUST_VALUE
RECENT_PURCHASES
ACCOUNT_STATUS
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE
WEB_DESTINATIONS
912
CUST_VALUE <7
910
ASK REASON FOR CALL
INPUT (NONE)
OUTPUT IVR_CHOICE
922
TRUE
920
CALCULATE_BUSINESS_VALUE_OF_CALL
INPUT:
IVR_CHOICE
WEB_DESTINATIONS
FRUSTRATION SCORE
MARKETING_VS_COLLECTIONS
LATE_PAYMENTS_SCORE
NET_PROFIT_SCORE
OUTPUT:
BUSINESS_VALUE_OF_CALL
932
NET_PROFIT_SCORE > 1000 AND CUST_REC.LAST_SENT_BONUS_CHECK < NOW - 60 AND MARKETING_VS_COLLECTIONS = "MARKETING"
OR
NET_PROFIT_SCORE > 500 AND FRUSTRATION_SCORE > 8 AND CUST_REC.LAST_SENT_BONUS_CHECK < NOW - 90 AND MARKETING_VS_COLLECTIONS = TRUE
930
CALCULATE_SEND_BONUS_CHECK
INPUT
CUST_REC
OUTPUT
SEND_BONUS_CHECK
942
TRUE
940
CALCULATE_CALL_PRIORITY
INPUT:
BUSINESS_VALUE_OF_CALL
FRUSTRATION_SCORE
OUTPUT:
CALL_PRIORITY
952
TRUE
950
CALCULATE_SKILL
INPUT:
BUSINESS_VALUE_OF_CALL
MARKETING_VS_COLLECTIONS
IVR_CHOICE
DNIS
WEB_DESTINATIONS
OUTPUT
SKILL
962
DISABLE IF BUSINESS_VALUE > 100 OR FRUSTRATION_SCORE > 5
960
CALCULATE_ON_QUEUE_PROMO
INPUT:
PROMO_HIT_LIST
OUTPUT
ON_QUEUE_PROMO
CALL_PRIORITY
SKILL
ON_QUEUE_PROMO
SCREEN_POP_LIST

FIG. 9

```
Module:  routing_decisions

      Submodule of:     routing_to_skill

      Input attributes: ANI
                        DNIS
                        account_number
                        cust_rec
                        cust_value
                        recent_purchases
                        frustration_score
                        late_payments_score
                        web_destinations

      Output attributes:call_priority : [1..4] \\corresponds to "low",
                               "med", "high", "top"
                        skill : ("norm_tier_dom", "norm_tier_intl",
                               "australia_promo", "high_tier",
                                    collections")
                        on_queue_promo : message_identifier
                        screen_pop_list : list ( screen_entry )

   Enabling condition:  true

   Type:          declarative

   Side-effect:         yes
```

Fig. 10

```
Module:  calculate_wrap_up

   Submodule of:  routing_to_skill

   Input attributes:    Ani
                        dnis
                        Web_DB_Load
                        Promos_Of_The_Day
                        Cust_Rec
                        Home_Phone
                        Account_Number
                        Cust_Value
                        Frustration_Score
                        Late_Payments_Score
                        Recent_Purchases
                        Marketing_VS_Collections
                        Web_Destinations
                        Call_Priority
                        Skill
                        On_Queue_Promo
                        Screen_Pop_List
                        Promo_Hit_List

   Output attributes:   wrap_up : set ( tuple ( att_name: string,
                                                value: string ))
   Enabling condition:  true

   Type:            decision

   Computation:
      Rules:            if true then wrap_up <- (att_name: "DNIS",
                                  value : convert-to-string (DNIS))
                        if true then wrap_up <- (att_name: "ANI",
                                  value: convert-to-string (ANI))
                        if true then wrap_up <- (att_name: "skill",
                                  value: skill)
                        if web_destinations not empty then wrap_up <-
                                  (att_name: \"web_destinations",
                                  value: (convert-to-string
                                         (web_destinations))
                        if cust_rec.card_color = "gold" <-
                                 (att_name:"frustration_score",
                                  value: convert-to-string
                                  (frustration_score))
      Combining policy:      wrap-up-cp //use contributions of all
                                         rules with true condition
      Side-effect:           yes

      Side-effect function:  write_into_archive ( wrap_up )
```

Fig. 11

```
Module:  get_recent_contacts_for_this_customer

   Submodule of:  info_about_customer

   Input attributes:    account_number

   Output attributes:   recent_contacts : list ( tuple ( date: date,
                                          event: event_type,
                                          delay_during_contact: int,
                                               \\ minutes
                                          delay_before_shipment: int
                                               \\ days
                                          amount: $value ) )
   Enabling condition:  true

   Type:                foreign

   Computation:         using recent_contacts_db
                        select date,event,amount
                        from contact_db
                        where acct_num = account_number

   Side-effect:         no
```

Fig. 12

```
Module:  get_recent_purchases_for_this_customer

   Submodule of:  info_about_customer

   Input attributes:    account_number

   Output attributes:   recent_purchases : list ( tuple ( date: date,
                                              item : 20digit,
                                              qty : int,
                                              amount : $value ) )
   Enabling condition:  true

   Type:           foreign

   Computation:         using purchase_db
                        select date,item,qty,amount
                        from purchases
                        where acct_num = account_number

   Side-effect:         no
```

Fig. 13

```
Module:  get_account_history_for_this_customer

    Submodule of:  info_about_customer

    Input attributes:    account_number

    Output attributes:   account_history : tuple ( overdue_amount:
                                                         $value,
                                                    number_days_overdue:
                                                         int,
                                                    history: list ( tuple (
                                                         date: date,
                                                    item : 20digit,
                                                    amount : $value )))
    Enabling condition:  true

    Type:           foreign

    Computation:         using account_history_db
                         select over_amt, num_days,history
                         from account_history
                         where acct_num = account_number

    Side-effect:         no
```

Fig. 14

```
Module: calculate_frustration_score

    Submodule of:  info_about_customer

    Input attributes:     recent_contacts

    Output attributes:    frustration_score : [1..10]

    Enabling condition:   VAL(recent_contacts)

    Type:           decision

    Computation:
       Rules:             if recent_contacts#1 defined then
                          frustration_score <-
                                      (value/50) *
                                      [(delay_during_contact/2) +
                                      max(0,delay_before_shipment -
                                      10)/3]

                          if recent_contacts#2 defined then
                          frustration_score <-
                                      (value/100) *
                                      [(delay_during_contact/2) +
                                      max(0,delay_before_shipment -
                                      10)/3]


       Combining policy:  frustration-score-cp //add contributions
                                                 of true rules and
                                                 round up, to max
                                                 of 10

     Side-effect:         no
```

Fig. 15

```
Module:  calculate_net_profit_score

   Submodule of:  info_about_customer

   Input attributes:    recent_contacts,
                        recent_purchases,
                        account_history,
                        cust_rec

   Output attributes:   net_profit_score

   Enabling condition:  recent_purchases#1.date<=now-60

   Type:           decision

   Computation:
      Rules:                      if recent_purchases not empty then
                                  net_profit_score <-
                                  10% * sum (recent_purchases#i.amount
                                  where recent_purchases#i.date > now -
                                  60)

                                  if recent_contacts not empty then
                                  net_profit_score <-
                                  -( 5 * count ( recent_contacts#i
                                  where recent_contacts#i.type =
                                  "complaint"))

                                  if account_history.overdue_amount > 0
                                  then net_profit_score <-
                                  - account_history.overdue_amount *
                                  account_history.number_days_overdue / 30

                                  if cust_rec.card_color = "platinum" then
                                  net_profit_score <- 100

                                  if cust_rec.card_color = "gold" then
                                  net_profit_score <- 50

                                  if cust_rec.card_color = "green" then
                                  net_profit_score <- 10

                                  if DISABLED(cust_rec) then
                                  net_profit_score <- 20

         Combining policy:        net-profit-score-cp //add contributions
                                                      of rules with true
                                                      conditions


    Side-effect:         no
```

Fig. 16

```
Module:  calculate_late_payment_score
   Submodule of:  info_about_customer
   Input attributes:    account_history
   Output attributes:   late_payment_score
   Enabling condition:  VAL(account_history)
   Type:          decision
   Computation:
      Rules:              if cust_rec.card_color = "platinum" then
                          late_payments_score <-
                          (account_history.overdue_amount
                          number_of_days_overdue)/100

                          if cust_rec.card_color = "gold" then
                          late_payments_score <-
                          (account_history.overdue_amount *
                          number_of_days_overdue)/50

                          if cust_rec.card_color = "green" then
                          late_payments_score <-
                          (account_history.overdue_amount *
                          number_of_days_overdue)/10

   Combining policy:    late-payment-score-cp //rule with true
                                                condition wins;
                                                default is 0

   Side-effect:         no
```

Fig. 17

```
Module:  calculate_cust_value

   Submodule of:  info_about_customer

   Input attributes:    net_profit_score,
                        late_payments_score,
                        cust_rec

   Output attributes:   cust_value

   Enabling condition:  true

   Type:          decision

   Computation:
      Rules:            if VAL(net_profit_score) then cust_value <-
                                    (1 - 1/net_profit_score) * 75

                        if cust_rec.card_color = "platinum" then
                        cust_value <- 20

                        if cust_rec.card_color = "gold" then cust_value
                        <- 10

                        if cust_rec.card_color = "green" then
                        cust_value <- 5

                        if VAL(frustration_score) then cust_value ←
                        5*frustration_score

      Combining policy: calculate-cust-val-cp //Add values of true
                                               rules and round up, to
                                               max of 100, default is
                                               0


   Side-effect:         no
```

Fig. 18

```
Module:  calculate_marketing_vs_collections

   Submodule of:  info_about_customer

   Input attributes:    cust_value,
                        late_payments_score

   Output attributes:   marketing_vs_collections

   Enabling condition:  late_payments_score > 0

   Type:          decision

   Computation:
      Rules:            if late_payments_score > f(cust_value) then
                        marketing_vs_collections <- "collect"
                        // f is function from [1..100] into [1..10],
                        // it could be linear, i.e., f(cust_value) =
                        //  cust_value/10
                        // or it could be shallower in beginning and
                           steeper
                        // towards end



      Combining policy :      marketing-vs-collection-cp //default is
                                                          "marketing",
                                                          any rule
                                                          with true
                                                          condition
                                                          wins

   Side-effect:         no
```

Fig. 19

```
Module: Ask_Reason_For_Call

   Submodule of:  routing_decisions

   Input attributes:      < none >

   Output attributes:     IVR_choice

   Enabling condition:    cust_value < 7 and DNIS not =
                          "Australia_promotion"

   Type:            foreign

   Computation:           x := IVR_dip( question(2));
                          if x = 1 then IVR_choice := "dom";
                          else if x = 2 then IVR_choice := "intl";
                          else IVR_choice[state] = EXC and
                          IVR_choice[EXC]=1

      Side-effect:              yes

      side-effect-function: IVR_dip( question(2))
```

Fig. 20

```
Module: calculate_business_value_of_call

   Submodule of:  routing_decisions

   Input attributes:     IVR_choice,
                         web_destinations,
                         frustration_score,
                         marketing_vs_collections,
                         late_payments_score,
                         net_profit_score

   Output attributes:    business_value_of_call : int

   Enabling condition:  true

   Type:           decision

   Computation:
      Rules:
                   if true then business_value_of_call <-
                        (cust_value/50 * net_profit_score)

                   if true then business_value_of_call <-
                        10*frustration_score

                   if DNIS = "Australia_promotion" then
                        business_value_of_call <- 100

                   if "Australia" in web_destinations[i].regions for
                        some i where
                        web_destinations[i].date_last_modified > now -
                        30
                        then business_value_of_call <- 100

                   if IVR_choice = "intl" then business_value_of_call <-
                        50

                   if marketing_vs_collections = "collect" then
                        business_value_of_call <-
                        (late_payments_score *
                        account_history.overdue_amount)/5

  Combining policy: business-value-of-call-cp // Add contributions of
                                                 rules with true
                                                 conditions and round up,
                                                 default is 0


    Side-effect:         no
```

Fig. 21

```
Module: Calculate_send_bonus_check
   Submodule of:  routing_decisions
   Input attributes:    cust_rec
   Output attributes:   send_bonus_check?
   Enabling condition:  if net_profit_score > 1000
                        and cust_rec.last_sent_bonus_check < now - 60
                        and marketing_vs_collections = "market"
                        OR
                        if net_profit_score > 500
                         and frustration_score > 8
                        and cust_rec.last_sent_bonus_check < now - 60
                         and marketing_vs_collections = "market"

 Type:              foreign
 Side-effect:             yes
      side-effect-function:
            issue_and_send_check($50,cust_rec.name,cust_rec.address)
```

Fig. 22

```
Module: call_priority

   Submodule of:  routing_decisions

   Input attributes:    business_value_of_call
                        frustration_score

   Output attributes:   call_priority

   Enabling condition:  true

   Type:          decision

   Computation:
      Rules:            if business_value_of_call < 25 then
                             call_priority <- 1

                        if 25 =< business_value_of_call < 100  then
                             call_priority <- 2

                        if 100 =< business_value_of_call < 500 then
                             call_priority <- 3

                        if 500 =< business_value_of_call then
                             call_priority <- 4

                        if frustration_score > 8 then
                             call_priority <- 4

                        if 6 =< frustration_score <= 8 then
                             call_priority <- 3

      Combining policy: call-priority-cp // high value wins with
                                            default result 2

    Side-effect:        no
```

Fig. 23

```
Module: calculate_skill

    Submodule of:    routing_decisions

    Input attributes:       business_value_of_call
                            marketing_vs_collections
                            IVR_choice
                            DNIS
                            web_destinations

    Output attributes:      skill

    Enabling condition:     true

    Type:               decision

    Computation:
        Rules:          if marketing_vs_collections = "collections"
                            then skill <- ["collections", infinity]

                        if business_value_of_call > 100
                            then skill <- ["high_tier", 40]

                        if DNIS = "australia_promotion" then
                            skill <- ["australia_promo", infinity]

                        if "Australia" in web_destinations[i].regions
                            for some i where web_destinations[i].date_last_modified >
                            now - 30 then
                            skill <- ["australia_promo", 20]

                        if cust_rec.estimated_income_bracket = ">100K-150K" then
                            skill <- ["australia_promo", 25]

                        if cust_rec.estimated_income_bracket = ">150K" then
                            skill <- ["australia_promo", 35]

                        if IVR_choice = "dom" then skill <- ["norm_tier_dom",30]

                        if IVR_choice = "intl" then skill <- ["norm_tier_intl",30]

                        if "US" in web_destinations[i].regions for some
                            i where web_destinations[i].date_last_modified >
                            now - 30 then
                            skill <- ["norm_tier_dom", 20]

                        if "US" not in web_destinations[i].regions for
                            some i where web_destinations[i].date_last_modified > now -
                            30 then
                            skill <- ["norm_tier_intl", 20]

        Combining policy: skill-cp //weighted sum policy, and ties are
                                broken by ordering "collections",
                                "australia_promo", "high_tier",
                                 "low_tier_intl", "low_tier_dom",
                                 default is ⊥

     Side-effect: no
```

Fig. 24

```
Module: calculate_on_queue_promo

   Submodule of:  routing_decisions

   Input attributes:    promo_hit_list

   Output attributes:   on_queue_promo

   Enabling condition:  DISABLE if business_value > 100 or

frustration_score > 5

   Type:          decision

   Computation:
      Rules:              if 60 < ACD.expected_wait_time(skill)
                               then on_queue_promo <-
                               promo_hit_list[#1]

                          if business_value_of_call < 30
                               then on_queue_promo <- promo_hit_list[#1]

      Combining policy: on-queue-promo-cp // first true wins, default
                                             is 0


      Side-effect:        no
```

Fig. 25

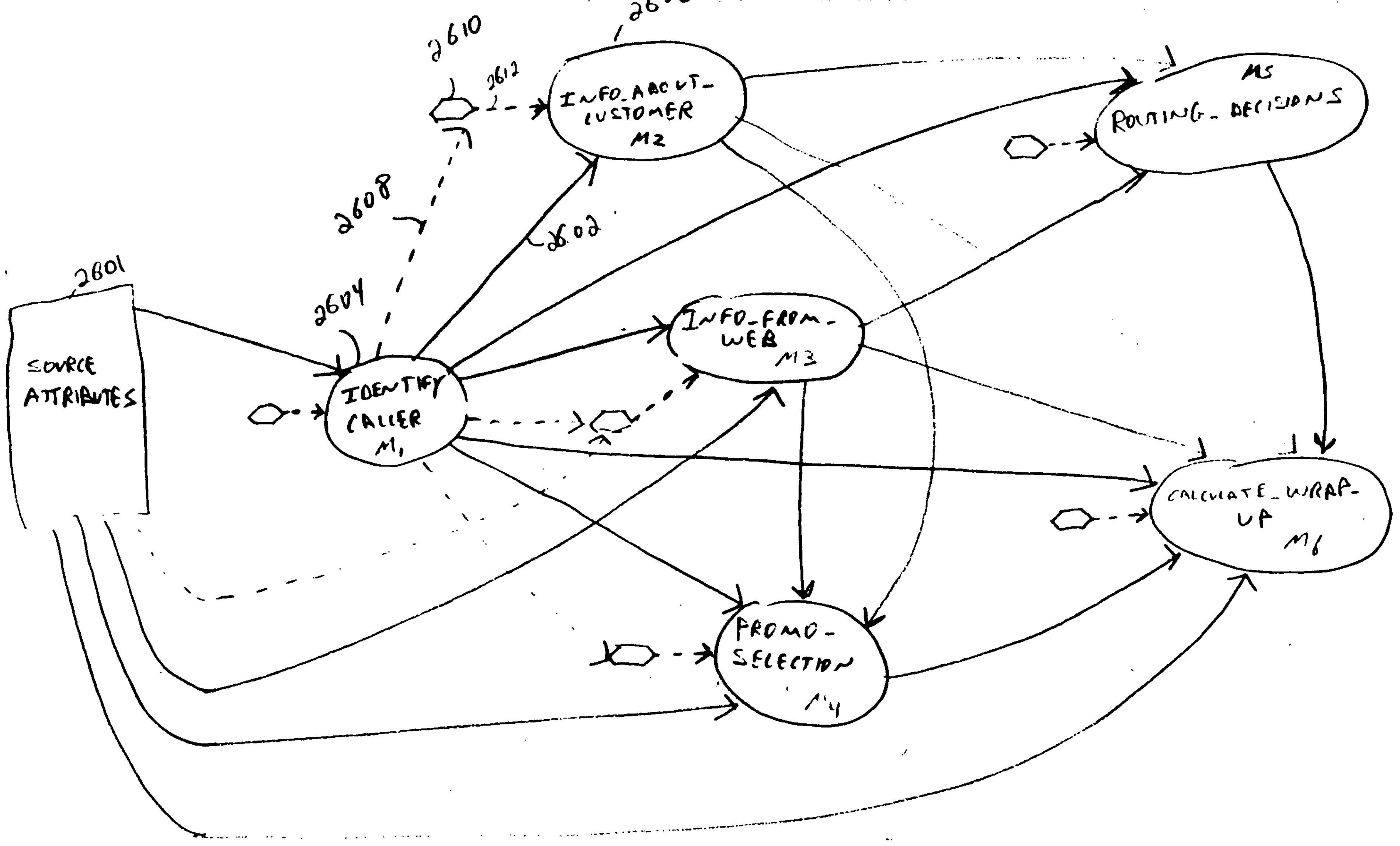

2606
2610
2612
2608
2602
2801
2604
SOURCE ATTRIBUTES
IDENTIFY CALLER M1
INFO_ABOUT_CUSTOMER M2
INFO_FROM_WEB M3
PROMO_SELECTION M4
M5 ROUTING_DECISIONS
CALCULATE_WRAP_UP M6

FIG 26

| | |
|---|---|
| $\dfrac{\sigma\vdash e:t}{\sigma\vdash \mathsf{value}(e):bool}$ | value |
| $\dfrac{\sigma\vdash f:AM:t_1\times\cdots\times t_n\rightarrow t,\sigma\vdash e_1:t_1,\cdots\sigma\vdash e_n:t_n}{\sigma\vdash \mathsf{Apply}(\langle f,e_1,\cdots,e_n\rangle):t}$ | apply |
| $\dfrac{\sigma\vdash e_1:t_1,\cdots\sigma\vdash e_n:t_n}{\sigma\vdash\langle e_1,\cdots,e_n\rangle:\langle a_1:t_1,\cdots,a_n:t_n\rangle}$ | tupling |
| $\dfrac{\sigma\vdash e_1:t,\cdots,\sigma\vdash e_n:t}{\sigma\vdash\{e_1,\cdots,e_n\}:\{t\}}$ | bagging |
| $\dfrac{\sigma\vdash e_1:t,\cdots,\sigma\vdash e_n:t}{\sigma\vdash[e_1,\cdots,e_n]:[t]}$ | listing |
| $\dfrac{\sigma\vdash e:\{t\}}{\sigma\vdash \mathsf{unitval}(e):t}$ | unitval |
| $\dfrac{\sigma\vdash e:\langle a_1:t_1,\cdots,a_n:t_n\rangle}{\sigma\vdash e.a_i:t_i}$ | projection on tuples |
| $\dfrac{\sigma\vdash e:[t]}{\sigma\vdash e\#i:t}$ | projection on lists |
| $\dfrac{\sigma\vdash e_1:[t_1],\sigma\vdash e_2:t_2}{\sigma\vdash \mathsf{factor}(e_1,e_2):[\langle f\_a:t_1,s\_a:t_2\rangle]}$ | factor (on lists) |
| $\dfrac{\sigma\vdash e_1:\{t_1\},\sigma\vdash e_2:t_2}{\sigma\vdash \mathsf{factor}(e_1,e_2):\{\langle f\_a:t_1,s\_a:t_2\rangle\}}$ | factor (on bags) |
| $\dfrac{\sigma\vdash f:t_1\rightarrow t,\sigma\vdash S:[t_1]}{\sigma\vdash \mathsf{map}(f)(S):[t]}$ | map (on lists) |
| $\dfrac{\sigma\vdash f:t_1\rightarrow t,\sigma\vdash S:\{t_1\}}{\sigma\vdash \mathsf{map}(f)(S):\{t\}}$ | map (on bags) |
| $\dfrac{\sigma\vdash id_\theta:t,\sigma\vdash\theta:t\times t\rightarrow t,\sigma\vdash S:\{t\}}{\sigma\vdash \mathsf{collect}(id_\theta,\theta)(S):t}$ | collect (on bags) |
| $\dfrac{\sigma\vdash id_\theta:t,\sigma\vdash\theta:t\times t\rightarrow t,\sigma\vdash S:[t]}{\sigma\vdash \mathsf{collect}(id_\theta,\theta)(S):t}$ | collect (on lists) |

FIG. 27

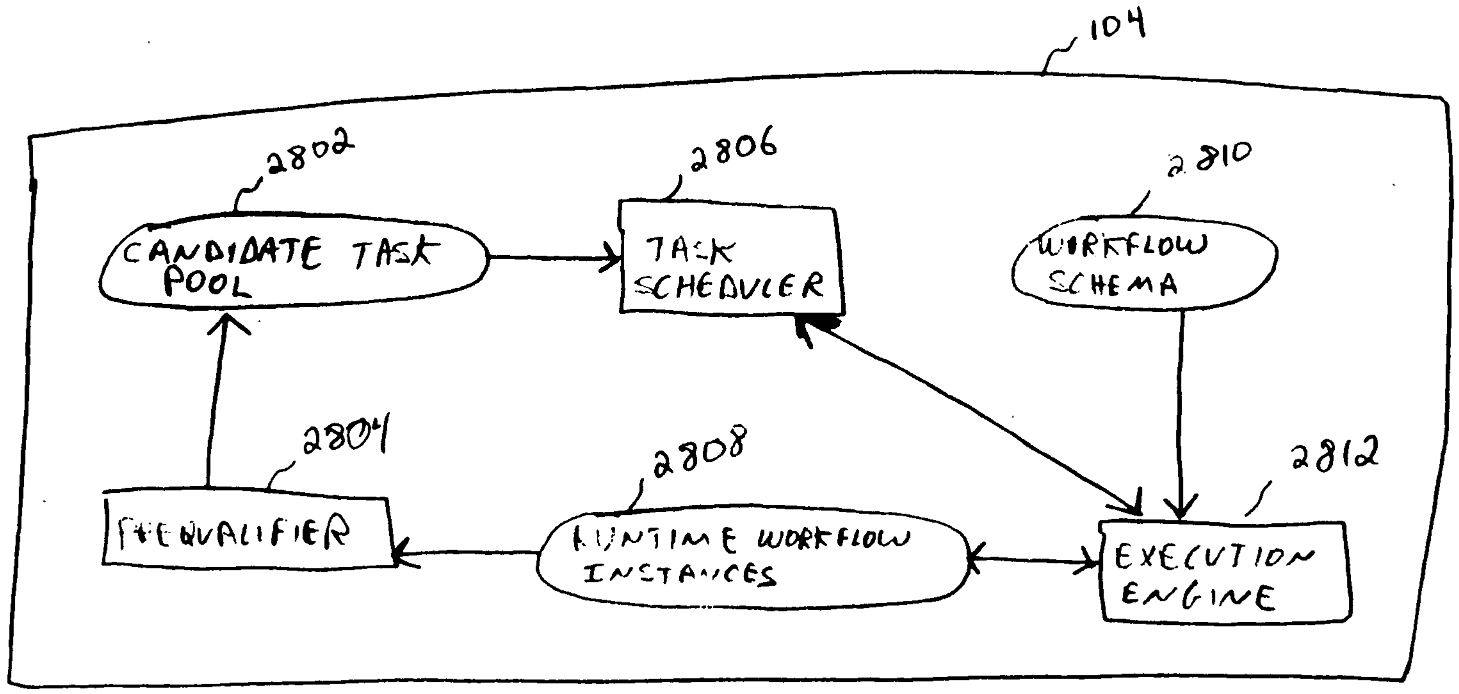

104
2802
CANDIDATE TASK POOL
2806
TASK SCHEDULER
2810
WORKFLOW SCHEMA
2804
PREQUALIFIER
2808
RUNTIME WORKFLOW INSTANCES
2812
EXECUTION ENGINE

FIG. 28

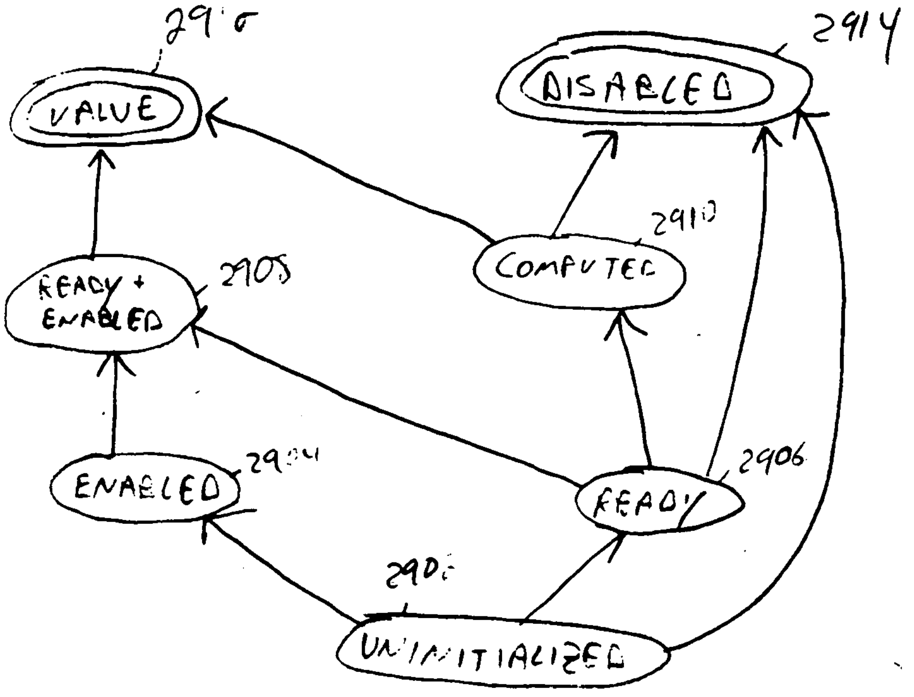

2916
VALUE
2914
DISABLED
2910
COMPUTED
READY + ENABLED
2908
ENABLED
2904
READY
2906
2902
UNINITIALIZED

FIG 29

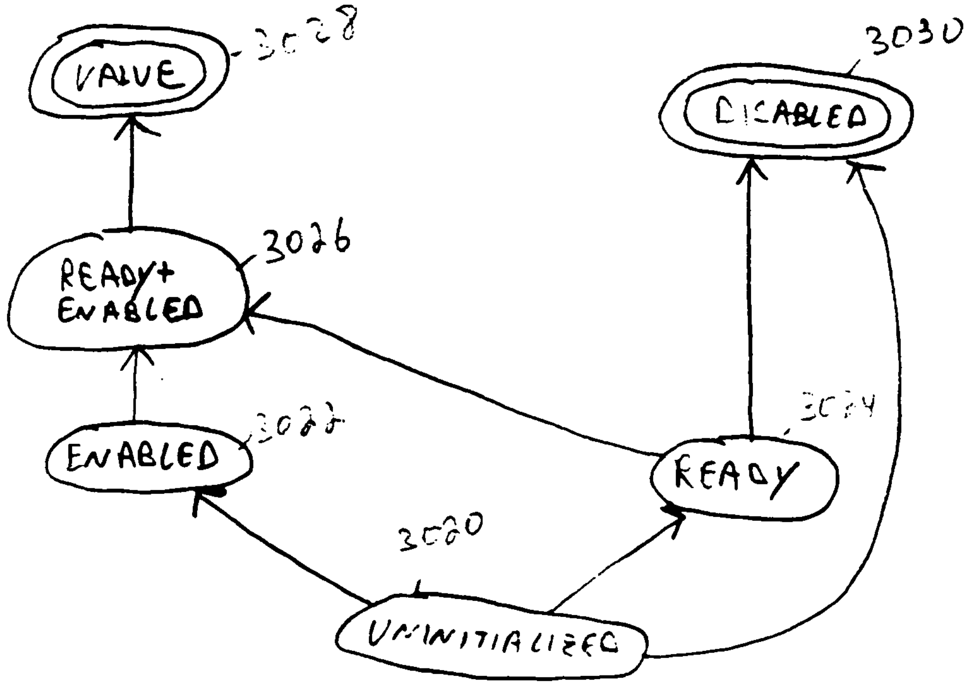

VALUE
3028
3030
DISABLED
READY+ ENABLED
3026
ENABLED
3022
READY
3024
3020
UNINITIALIZED

FIG 30

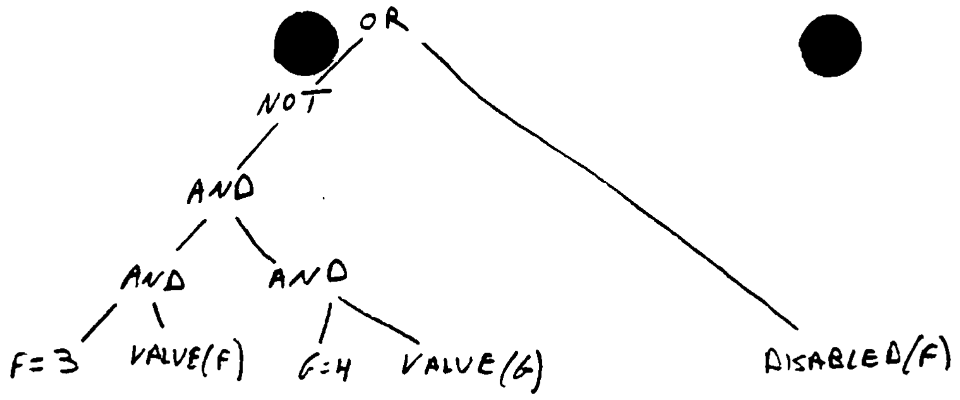
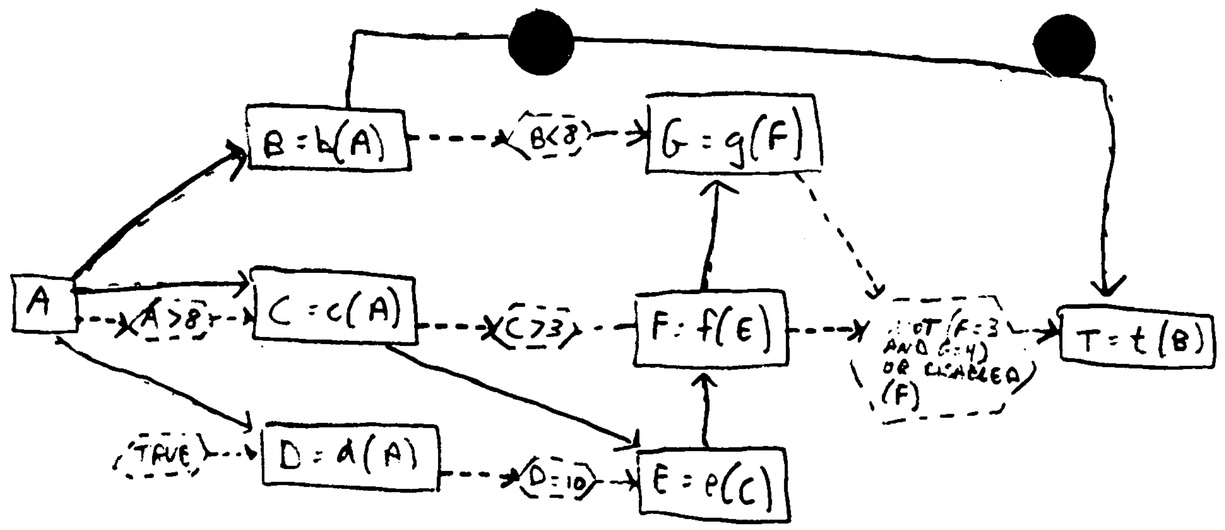
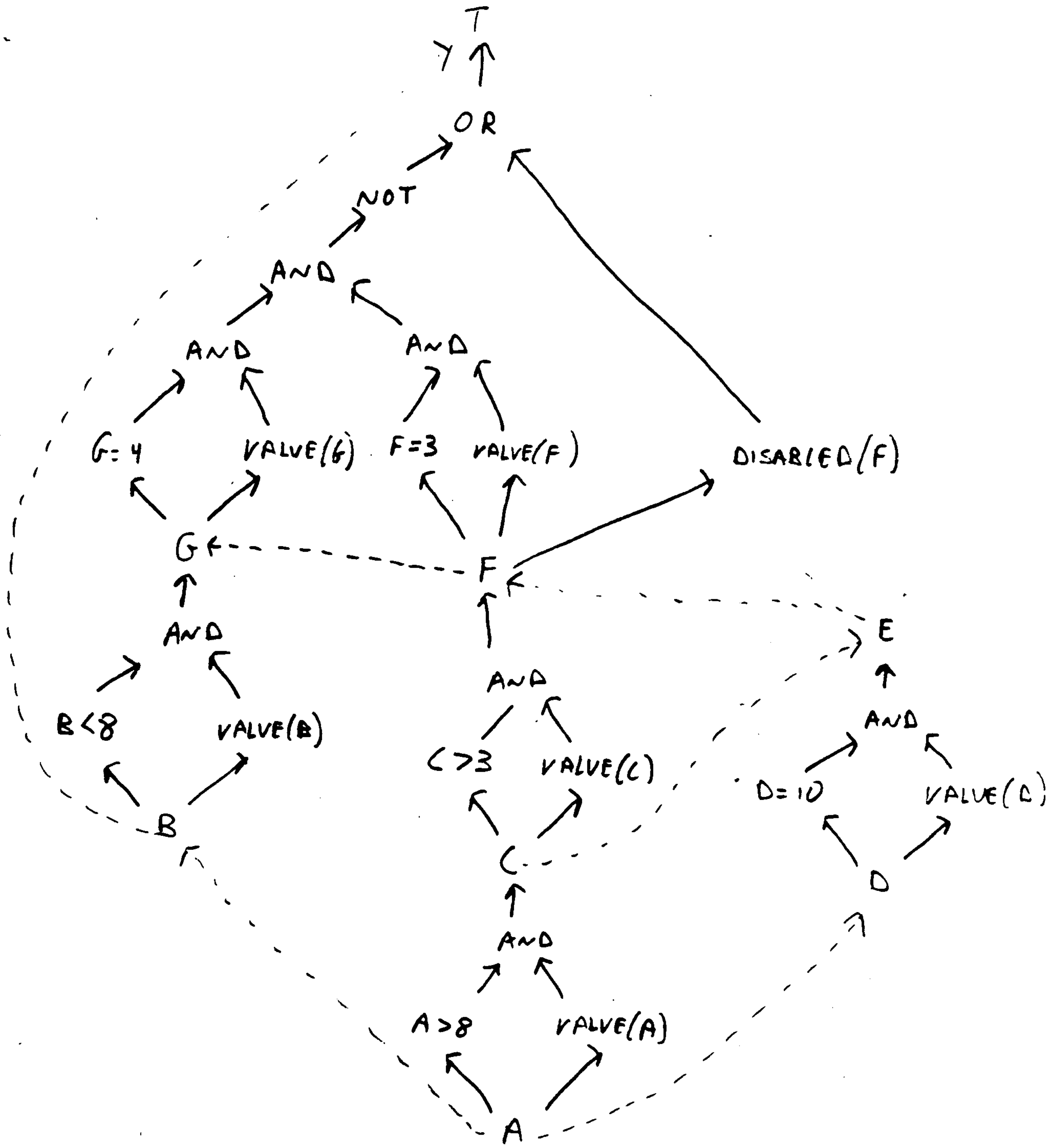


FIG. 31



A
B = b(A)
C = c(A)
D = d(A)
TRUE
A > 8
B < 8
C > 3
D = 10
G = g(F)
F = f(E)
E = e(C)
NOT (F = 3 AND G = 4) OR DISABLED (F)
T = t(B)

FIG. 32



T
Y
OR
NOT
AND
AND
AND
G=4
VALVE(G)
F=3
VALVE(F)
DISABLED(F)
G
F
E
AND
AND
AND
B<8
VALVE(B)
C>3
VALVE(C)
D=10
VALVE(D)
B
C
D
AND
A>8
VALVE(A)
A

FIG. 33

*Global variables:*

```
These variables are global to the whole execution of workflow instance
    G: a dependency graph
    S: set of source attribute nodes of G
    T: set of target attribute nodes of G
    σ []: array of attribute states
    μ []: array of attribute values
    α []: array of three valued logic values (true, false unknown)
    HIDDEN_EDGE: set of hidden edges of G.
    HIDDEN_ATT: set of hidden attribute nodes of G.
```
3402

*Notations:*

```
    σ [A]: element of array σ [] that corresponds to the attribute node A in G
    μ [A]: element of array μ [] that corresponds to the attribute node A in G
    α [p]: element of array α [] that corresponds to the condition node p in G
```
3404

```
Initialization phase:
    procedure Init:
    Input:
        g: a dependency graph:
        So: source nodes in g
        Te: terminal nodes in g
    body:
    BEGIN init
        G:=g ; S : = So: T : = Te;
        /*Initialization of the states and values of attributes nodes */
        FOR all the attribute nodes A in G DO
            IF A ∈ S    /* A is a source node */
                THEN σ [A] := READY + ENABLED
                ELSE σ [A] := UNITIALIZED;
            μ [A] := NULL;
        END FOR
        /* Initialization of α-values of condition nodes */
        FOR all the condition nodes p in G DO
            α[A] := unknown;
        END FOR
        */ Initialization of the set of hidden edges and hidden nodes */
        HIDDEN_EDGE := ø; HIDDEN_ATT := ø
    END init
```
3406

3408

3410

3412

FIG 34A

```
Increment                                                          3414
  Input:
      A : an attribute in G.   3416
      v : a value for A.
  body:
     BEGIN increment
  μ[A] : = v;                  3418
  IF σ [A] = READY                                      3420
       THEN propagate_att_change(A, COMPUTED)
  IF σ [A] = READY+ENABLED
       THEN propagate_att_change(A, VALUE)
END Increment
                                                                                3422
propagate_att_change
  Input:
      B : an attribute in G.   3424
      σ : a state for B
  body:
 /* Set state for B*1
IF ((σ[B] = ENABLED) AND (σ = READY)) OR (σ [B] = READY) AND (σ =ENABLED))   3426
       THEN σ [B] := READY+ENABLED
       ELSE σ [B] := σ;                                                       3428
/* push relevant information to the affected successor nodes */              3430
CASE : σ [B] ∈ {VALUE, COMPUTED}   /* The value of B is computed */
  /* try to evaluate predicate nodes that are using the value of B */
  FOR each condition node p of the form pred(t_1, ⋯, t_n) such that (B,p) ∈ G DO   3432
       IF (B,p) ∉ HIDDEN_EDGE   3434
            THEN
               Hide_edge ((B,p));   3436                                   3438
               IF Eval (p) ≠ unknown THEN α [p] := Eval(p); propagate_cond_change(p)
  END FOR
/* check if the attributes nodes that have B as input parameters are READY */
FOR each attribute node C such that (B, C) ∈ G DO
IF σ[B]=VALUE THEN                                                           3440
       IF (B, C) ∉ HIDDEN_EDGE
            THEN
               Hide_edge((B,C));
               IF there exists no attribute node D such that (D, C) ∉ HIDDEN_EDGE
                   THEN propagate_att_change (C, READY);
END FOR
CASE : σ [B] = ENABLED
 /* evaluates condition nodes of the form VALUE (B) and DISABLED (B) */
 FOR each condition node p of the form VALUE (B) or DISABLED (B) such that (B,p) ∈ G DO
       IF (B,p) ∉ HIDDEN_EDGE

                                                                          3442
```

FIG 34B

```
            THEN
              Hide_edge((B,p))
              IF p is of the form VALUE (A) THEN α[p] := true ELSE α[p]: = false;   3442
              propagate_cond_change(p);
         END FOR                                                                     3444
CASE : σ [B] = DISABLED
    /* evaluate condition nodes of the form VALUE (B) and DISABLED (B) */
    FOR each condition node p of the form VALUE (B) or DISABLED (B) such that (B,p) ∈ G DO
      IF (B,p) ∉ HIDDEN_EDGE
        THEN
          Hide_edge ((B,p));
          IF p is of the form VALUE (A) THEN α[p] := false ELSE α[p] := true;
          propagate_cond_change(p);
    END FOR
    /* check if the attribute nodes that have B as input parameters are READY */
    FOR each attribute node C such that (B,C) ∈ G DO
      IF (B,C) ∉ HIDDEN_EDGE
          THEN                                                                       3446
            Hide_edge((B,C));
            IF there are no more attribute nodes D such that (D,C) ∉ HIDDEN_EDGE
              THEN propagate_att_change (C,READY);
    END FOR                                                                          3438
    /* If the attribute is stable then hide the attribute */
    IF (σ [B] ∈ {DISABLED, VALUE}) THEN Hide_node(B);   3448
    END propagate_att_change
                                                                                     3422
    propagate_cond_change
        Input:                                                                       3450
          p: a condition node in G.
        body:
        BEGIN propagate_cond_change
        let n be the successor of p in G   3452                                      3454
        IF (p,n) ∉ HIDDEN_EDGE
          THEN
              Hide _edge ((p,n));   3456                                             3458
              CASE: n is OR condition node
3460          IF (α [p] = true) THEN α [n] : = true; propagate_cond_change(n); END IF;
              IF α [p] = false  AND for each condition node p' where (p',n) ∈ G, (p',n) ∈
3462          HIDDEN_EDGE
                   THEN α [n] : = false; propagate_cond_change(n);END IF;
              CASE: n is a AND node
3466          IF (α [p] = false) THEN α [n] := false; propagate_cond_change(n);END IF;
              IF α [p] = TRUE AND for each condition node p' where (p',n) ∈ G, (p',n) ∈
3468          HIDDEN_EDGE
                                                                                     3464
```

FIG. 34C

3454

```
3468 -      THEN α [n]: = TRUE ; propagate_cond_change(n);END IF;  - 3464
    CASE : n is NOT node
        α [n]: = ¬ (α [p]) ; propagate_cond_change(n);  } 3470
    CASE : n is an attribute node
        IF (α [p] = true)
            THEN propagate_att_change(n, ENABLED)  - 3472
            ELSE propagate_att_change(n,DISABLED);
  END propagate_cond_change
```

3452

```
Hide_edge
  Input
    (n,n') : an edge in G.
body
BEGIN Hide_edge
HIDDEN_EDGE : = HIDDEN_EDGE ∪ {(n,n')];
IF (there are no more edges (n, n") ∈ G such that (n,n") ∉ HIDDEN_EDGE  - 3474
    THEN Hide_node(n)
END Hide_edge
```

```
Hide_node
    Input
        n : a node in g.
    body
    BEGIN Hide_node
    HIDDEN_ATT := HIDDEN_ATT ∪ {n}  - 3476
    FOR each edge (n',n) ∈ g such that (n',n) ∉ HIDDEN_EDGE) DO
        Hide_edge (n',n)
     END FOR
  END Hide_node
```

FIG 34 D

*Global variables:*

*These variables are global to the whole execution of workflow instance*

*G* : a dependency graph
*S* : set of attribute nodes of *G* /* this set contains the source nodes */
*T* : set of attribute nodes of *G* /* this set contains target nodes */
$\sigma$[] : array of attribute states
$\alpha$[] : array of three valued logic values (*true, false unknown*)
*HIDDEN_EDGE* : set of edges of *G*.
*HIDDEN_ATT* : set of attribute nodes of *G*.

3504

*T_N*[][] : Matrix of integers that associates an integer value to each pair (*p*,*A*) where *p* is a condition node and *A* is an attribute node
in *G*
/* *T_N*[*p*][*A*] = 0 means that the attribute *A* is True_necessary for the the condition node *p**/

*F_N*[][] : Matrix of integers that associates an integer value to each pair (*p*;*A*) where *p* is a condition node and *A* is an attribute node in *G*
/**F_N*[*p*][*A*] = 0 means that the attribute *A* is False_necessary for the condition node *p**/

*V_N*[][] : Matrix of integers associates an integer value to each pair (*B*,*A*) where *B* and *A* are attribute nodes in *G*
/**V_N*[*B*][*A*] = 0 means that the attribute *A* is Value_necessary for the attribute node *B**/

*S_N*[][] : Matrix of integers associates an integer value to each pair (*B*,*A*) where *B* and *A* are attribute nodes in *G*
/**S_N*[*B*][*A*] = 0 means that the attribute *A* is Stable_necessary for the attribute node *B**/

*N*[] : Array of boolean
*N*[*A*] = *true* means that the attribute *A* is computed as necessary/*
*N*[*A*] = *false* means that the attribute *A* is not computed as necessary*/

35

*Notations* :

***nb_pred*(*p*)** : number of predecessors of *p* in *G*

```
Initialization phase:
  procedure Init :
  Input:
    g : a dependency graph:
    So : source nodes in g
    Te : terminal nodes in g
  body:
  BEGIN N_init
```

3506

FIG. 35A

```
Init() } 3508

/* Initialization of T_N,F_N,S_N,V_N */
FOR all the condition nodes $p$ in $G$ DO
   FOR all the attribute nodes $A$ in $G$ DO

      CASE : $p$ is an OR node :
         $T\_N[p][A]$ := $nb\_pred(p)$:          /* rule 1 */  } 3511
         $F\_N[p][A]$ := 1;                     /* rule 2 */

      CASE : $p$ is an AND node :
         $T\_N[p][A]$ := 1;                     /* rule 3 */
         $F\_N[p][A]$ := $nb\_pred(p)$;          /* rule 4 */

      CASE : $p$ is a NOT node :
         $T\_N[p][A]$ := 1;                     /* rule 5 */
         $F\_N[p][A]$ := 1;                     /* rule 6 */

      CASE : $p$ is a node of the form VAL($B$) or DIS($B$):
         $T\_N[p][A]$ := 1;                     /* rules 7 and 9 */
         $F\_N[p][A]$ := 1;                     /* rules 8 and 10 */

      CASE : $p$ is a node of the form $pred(t_1,\ldots t_n)$:
         $T\_N[p][A]$ := 1;                     /* rule 11 */
         $F\_N[p][A]$ := 1                      /* rule 12 */

   END FOR
END FOR

FOR all the attributes nodes $A$ in $G$ DO
   FOR all the attribute nodes $B$ in $G$ DO          } 3512
      $S\_N[A][B]$ := 1; $V\_N[A][B]$ := 1
   END FOR
END FOR

FOR all the attributes nodes $A$ in $G$ DO
   $N[A]$ := false                                     } 3513
END FOR

END N_init
```
3510

3506

**N_Increment**
*Input :*
$A$ : an attribute in $G$.
$v$ : a value for $A$.
*Variables* /* Global to one execution of the increment phase (for one execution step) */ } 3518

351

FIG 35B

```
prev_E: set of attribute nodes
        /* used to store the nodes that were READY+ENABLED or ENABLED (in a
        previous execution of N-increment) */

prev_HIDDEN_EDGE: /* set of edges*/
        used to store the edges that were previously hidden (in the previous steps) */

prev_T_N: set of pairs (p,A) where p is a condition node and A is an attribute node
        /* used to denote the elements of T_N[p][A] that were set to 0 in a previous
        execution of N-increment*/                                                    3518

prev_F_N: set of pairs (p,A) where p is a condition node and A is an attribute node
        /* used to denote the elements of F_N[p][A] that were set to 0 in a previous
        execution of N-increment*/

Δ_E: set of attribute nodes
  /* used to store the new ENABLED or READY+ENABLED attribute nodes that were
  neither ENABLED nor READY+ENABLED in the previous steps. */                         3520
Δ_HIDDEN_EDGE : set of edges
  /* used to store the new hidden edges */

new_V_N : set of pair (A,A) where A is an attribute node
  /* used to store the positions of elements of V_N[][] whose new value is zero due to
    case 1 */
new_S_N : set of pair (B,A) where B and A are attribute nodes
  /* used to store the positions of elements of S_N[][] whose new value is zero due to
    case 2 */
new_T_N : set of pair (p,A) where p is a predicate node and A is an attribute node    3522
  /* used to store the positions of elements of T_N[][] whose new value is zero due to
    some new hidden edges (case 3) */
new_F_N : set of pair (p,A) where p is a predicate node and A is an attribute node
  /* used to store the positions of elements of F_N[][] whose new value is zero due to  351
    some new hidden edges (case 4) */

body:
BEGIN N_increment

/* preparation step: */
prev_HIDDEN_EDGE := HIDDEN_EDGE;
prev_E := {A | A is an attribute node in G and σ [A] ∈ {READY+ENABLED,               3524
ENABLED}

Increment(A,v) } 3526

/* Instigation step : Compute new necessary properties according to the instigation     3528
cases*/
```

FIG 35C

```
Case 1 :                                                                              3530
  Δ_E := {A|A is an attribute node in G and σ[A] ∈ {READY+ENABLED, ENABLED}
  and A ∉ prev_E}
  new_V_N := Ø;
  FOR each attribute node A in Δ_E DO
    V_N[A][A] := 0; new_V_N := new_V_N U {(A,A)}/* a node is value_necessary for
    itself*/
  END FOR

Case 2 :                                                                              3532
  new_S_N := Ø;
  FOR each attribute node B in Δ_E DO
    FOR each attribute node in A in G such that σ [A] ∈ {READY+ENABLED,
    ENABLED} DO
      IF V_N[B][A] = 0 and S_N[B][A] = 1
          THEN S_N[B][A] = 0; new_S_N := new_S_N U {(B,A)} /*         rule (13)*/
     END FOR
   END FOR


  Δ_HIDDEN_EDGE := HIDDEN_EDGE – prev_HIDDEN_EDGE                                     3534
  prev_T_N := {(p,A) | T_N[p][A] =0 }
  prev_F_N := {(p,A) | F_N[p][A] =0 }
  new_T_N := Ø;
  new_F_N := Ø;


  FOR all edges (n,p) ∈ Δ_HIDDEN_EDGE such that p ∉ HIDDEN_ATT and p is a
  condition node DO
    FOR all attribute nodes A such that σ (A) ∉ {COMPUTED, VALUE, DISABLED}
    DO
        CASE: 3                                                                       3536
      CASE : p is an OR node:
        IF (n,A) ∉ prev_T_N
        THEN
           T_N[p][A] := T_N[p][A] –1;              /* rule (1)*/
           IF T_N[p][A] = 0 THEN new_T_N := new_T_N U {(p,A)}

        CASE: 4                                                                       3538
        CASE : p is an AND node :
        IF (n,A) ∉ prev_F_N /* same reasoning as for OR nodes but with rule 4*/
        THEN
           F_N[p][A] := F_N[p][A] –1;              /* rule (4)*/
           IF F_N[p][A] = 0 THEN new_F_N := new_F_N ∪ {(p,A)}                         352
    END FOR
  END FOR
```

FIG 35C

```
/* Propagation step */
   New_propagate(new_V_N, new_S_N, new_T_N, new_F_N)  3540
END N_Increment
                                                                    3542
New_propagate
   Input :
    new_V_N : set of pairs (A,A) where A is an attribute node
    new_S_N : set of pairs (B,A) where B and A are attribute nodes
    new_T_N : set of pairs (p,A) where p is a condition node in G and A is an attribute
    node
    new_F_N : set of pairs (p,A) where p is a condition node in G and A is an attribute
    node
   body:
   FOR each pair (A,A) in new_V_N DO
      propagate_V_N(A,A)
      FOR each attribute node B such that $(A,B) \in G$ and $(A,B) \notin$ HIDDEN_EDGE  3546
        V_N[B][A] := 0; propagate_V_N(B,A)/*      rule (16) */
      END FOR
   END FOR                                                          3544
   FOR each pair (B,A) in new_S_N DO
      propagate_S_N(B,A)
   END FOR
     FOR each pair (p,A) in new_T_N DO
       propagate_T_N(p,A)
   END FOR
   FOR each pair (p,A) in new_F_N DO
       Propagate_F_N(p,A)
   END FOR
 END N-propagate

 propagate_V_N                                                      3548
   Input :
     B : an attribute node in G.
     A : an attribute node in G./* A is newly Value_necessary for B*/
   body:
   IF σ[B] = ENABLED and S_N[B][A] = 1                              3550
      THEN S_N[B][A] = 0; propagate_S_N(B,A)          /*rule (13) */
  ELSE let p be the condition node such that $(p,B) \in G$.         3552
        IF F_N[p][A]=0 and S_N[B][A] = 1
          THEN S_N[B][A] = 0; propagate_S_N(B,A)            /*rule (14)*/
  END IF
  FOR each condition node p of the form pred($t_1 \cdots, t_n$)
        such that $(B,p) \in g$ and $(B,p) \notin$ HIDDEN_EDGE DO
   IF T_N[p][A] = 1                                                 3554
     THEN T_N[p][A] : = 0;propagate_T_N(p,A)         /*rule (11)*/
   IF F_N[p][A]  = 1
    THEN F_N[p][A] : = 0;propagate_F_N(p,A)          /*rule (12)*/  3556
```

FIG 35E

3542

3548

```
 END FOR
END propagate_V_N
```

3558

3560

```
propagate_S_N
  Input:
     B : an attribute node in G.
     A : an attribute node in G./* A is newly Stable_necessary for B*/
   body:
  FOR each attribute node C such that $(B,C) \in g$ and $(B,C) \notin HIDDEN\_EDGE$ DO
     IF V_N[C][A] = 1 THEN V_N[C][A] = 0;propagate_V_N(C,A)  /* Rule 17 */
  END FOR
  IF $B \in T$ THEN N[A] : = true
END propagate_S_N
```

3562

3564

```
propagate_F_N
   Input:
      p : a condition node in G.
      A: an attribute node in G./* A is newly False_necessary for p*/
   body:
      let n be the successor of p in G
      IF $(p,n) \in HIDDEN\_EDGE$
           THEN
             CASE : n is an OR or AND node
                IF F_N[n][A] > 0
                    THEN
                     F_N[n][A] := F_N[n][A] - 1;              /*rules (2) and (4)*/
                     IF F_N[n][A] = 0 THEN propagate F_N (n,A)
               CASE : n is a NOT node
                 IF T_N[n][A] = 1 THEN T_N[n][A] : = 0;propagate_T_N(n,A) /*rule (6)*/
                CASE : n is an attribute node
                  IF (T_N[p][A] = 0 or V_N[n][A] = 0 and S_N[n][A] = 1
                     THEN S_N[n][A] = 0;propagate_S_N(n,A)     /*rules (14) and (15)*/
                   FOR each condition node p' of the form VALUE (n)
                          such that $(n,p') \in g$ and $(n,p') \notin HIDDEN\_EDGE$ DO
               IF F_N[p'][A] = 1 THEN F_N[p'][A] : = 0;propagate_F_N(p',A) /*rule (8)*
             END FOR
             FOR each condition node p' of the form DISABLED (n)
                   such that $(n,p') \in G$ AND $(n,p') \notin HIDDEN\_EDGE$ DO
              IF T_N[p'][A] = 1 THEN (T_N[p'][A] := 0;propagate_T_N(p',A)  /*rule (10)*/
           END FOR
 END propagate_F_N
```

3566

```
 propagate_T_N
   Input:
     p: a condition node in G.
    A: an attribute node in G/*  A is newly True_necessary for p*/
    body:
```

FIG 35F

```
        let n be the successor of p in G
        IF (p,n) ∉ HIDDEN_EDGE
            THEN
              CASE : n is an OR or AND node
                IF T_N[n][A] > 0
                    THEN
                      T_N[n][A] := T_N[n][A] - 1;   /*rules (1) and (3)*/
                      IF T_N[n][A] = 0 THEN propagate_T_N(n,A)
              CASE : n is a NOT node
                IF F_N[n][A] = 1 THEN F_N[n][A] : = 0; propagate_F_N(n,A) /* rule (5) */
              CASE : n is an attribute node
                  IF F_N[p][A] = 0 and S_N[n][A] = 1
                      THEN S_N[n][A] = 0;propagate_S_N(n,A)  /*rule (15)*/
                  FOR each condition node p' of the form VALUE (n)
                        such that (n,p') ∈ G and (n,p') ∉ HIDDEN_EDGE DO
                     IF T_N[n][A] = 1 THEN
                           T_N[p'][A] : = 0;propagate_T_N(p',A)      /*rule (8)*/
                  END FOR
                  FOR each condition node p' of the for DISABLED (n)
                        Such that (n,p') ∈ G and (n,p') ∉ HIDDEN_EDGE DO
                     IF F_N[n][A] = 1 THEN
                      F_N[p'][A] : = 0;propagate_F_N(p',A)      /*rule (9)*/
                   END FOR
      END propagate_T_N
```

3566

3542

FIG 35G

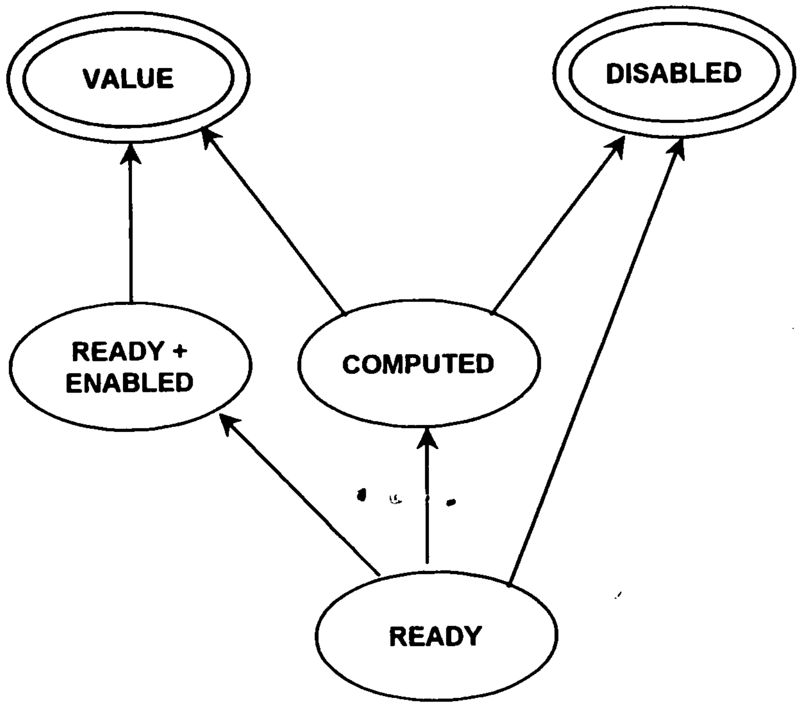


FIG 36

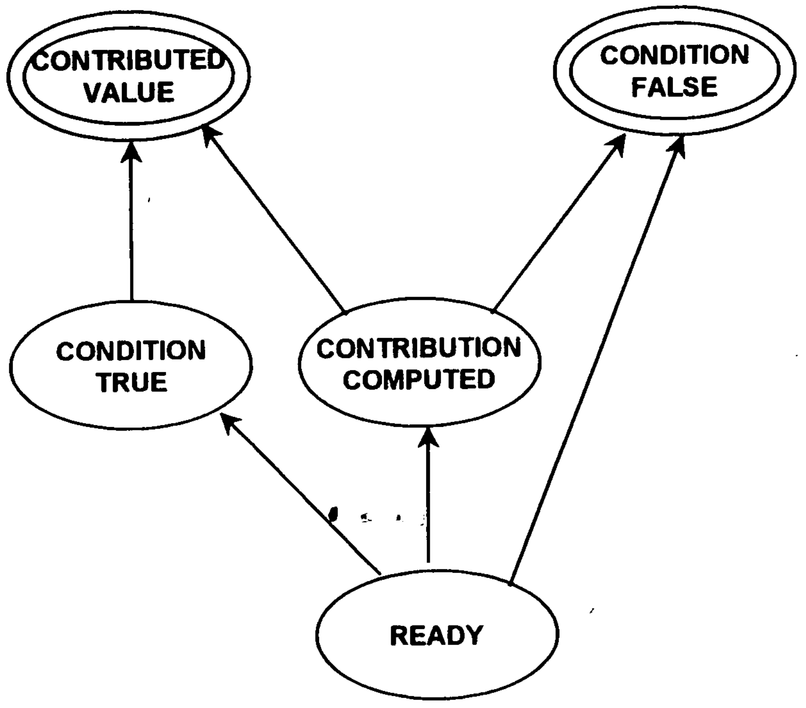

CONTRIBUTED VALUE
CONDITION FALSE
CONDITION TRUE
CONTRIBUTION COMPUTED
READY

FIG. 37

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | source | | get_recent_ contacts_... (node 504) | get_recent_ purchases_... (node 508) | get_account_ history_... (node 512) | calculate_ frustration_ score (node 516) | calculate_ net_profit_ score (node 520) | calculate_ late_payments_ score (node 524) | calculate_ cust_value (node 528) | calculate_ marketing_vs_ collections (node 532) |
| 2 | | | foriegn module | foriegn module | foriegn module | "add contribs. of true rules and round up, to max of 10" | "add contribs. of true rules" | "true rule wins; default is 0" | "add contribs. of true rules and round up, to max of 100" | "any true rule gives collect; default is marketing" |
| 3 | cust_rec | account_ number | recent_ contacts | recent_ purchases | account_ history | frustration_ score | net_profit_ score | late_ payment_ score | cust_ value | marketing_ vs_ collections |
| 4 | <"John Doe", "101 Ash, LA", "gold", FALSE SI ...> | 421136 SI | NS | NS | NS | NS | NS | NS | NS | NS |
| 6 | | | ENABLED READY | ENABLED READY | ENABLED READY | READY | READY | READY | ENABLED READY | READY |
| 6 | | | | | | READY | READY | ⊥ | READY | "collect" C-C |
| 7 | | | | | | READY | READY | CONDITION TRUE | ⊥ | |
| 8 | | | | | | | READY | ⊥ | 10 C-V | |
| 9 | | | | | | | ⊥ | | ⊥ | |
| 10 | | | | | | | 50 C-V | | READY | |

FIG. 38

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | source | | get_recent_ contacts_... (node 504) | get_recent_ purchases_... (node 508) | get_account_ history_... (node 512) | calculate_ frustration_ score (node 516) | calculate_ net_profit_ score (node 520) | calculate_ late_payments_ score (node 524) | calculate_ cust_value (node 528) | calculate_ marketing_vs_ collections (node 532) |
| 2 | | | foriegn module | foriegn module | foriegn module | "add contribs. of true rules and round up, to max of 10" | "add contribs. of true rules" | "true rule wins; default is 0" | "add contribs. of true rules and round up, to max of 100" | "any true rule gives collect; default is marketing" |
| 3 | cust_rec | account_ number | recent_ contacts | recent_ purchases | account_ history | frustration_ score | net_profit_ score | late_ payment_ score | cust_ value | marketing_ vs_ collections |
| 4 | <"John Doe", "101 Ash, LA", "gold", FALSE ... > SV | 421136 SV | NS | [<8-10-98,coat, 1, $50> <6-15-98,hat, 1, $20 > SV | <10,45,[<9-18-98 pay,$40 > <8-10-98, order,$50>] SV | NS | ⊥ SU | 9 SV | NS | NS |
| 5 | | | ENABLED + READY | VALUE | VALUE | READY | DISABLED | VALUE | ENABLED + READY | ENABLED + READY |
| 6 | | | | | | READY | ⊥ | ⊥ | ⊥ | "collect" C-C |
| 7 | | | | | | READY | READY | 9 C-V | ⊥ | |
| 8 | | | | | | | -9 C-V | ⊥ | 10 C-V | |
| 9 | | | | | | | ⊥ | | ⊥ | |
| 10 | | | | | | | 50 C-V | | READY | |

FIG. 39

Initialization

```
Based on the DL specification, compute rows 1, 2, and 3 of the display; 4002
For source attribute cells of row 4 do:
    For each source attribute with value, insert value and apply
"attribute_value_indication";
    For each source attribute that is disabled, apply
"attribute_disabled_indication";                                        4004
 For each non-decision module
     In row 5, apply "module_uninitialized_indication";
     In row 4, apply "attribute_uninitialized_indication";              4006
 For each decision module
     In row 5, apply "module_ready_indication";
     In row 4, apply "attribute_uninitialized_indication";              4008
 For each cell in rows 6,7,8,., apply "rule_ready_indication"            4010
```

Iteration

```
For each event of execution engine do
Case on event_type
    non_dec_module_enabled:
          in row 5, apply "module_enabled_indication";                  4012

    non_dec_module_ready:
          in row 5, apply "module_ready_indication";                    4014

    non_dec_module_ready+enabled:
          in row 5, apply "module_ready+enabled_indication";            4016

    non_dec_module_computed::
          in row 5, apply "module_computed_indication";
          in row 4, label corresponding attribute cell with the value computed   4018
 and apply
                 "attribute_computed_indication";

    non_dec_module_value:
          in row 5, label cell for this module as "value" and apply
 "module_value_indication";
          in row 4, label corresponding attribute cell with value assigned and   4020
 apply
                 "attribute_value_indication"

    non_dec_module_disabled:                                             4022
```

FIG 40A

```
                in row 5, label cell for this module as "disabled" and apply
                        "module_disabled_indication";
                in row 4, label corresponding attribute cell with "⊥" and apply
                        "attribute_disabled_indication"

        dec_module_enabled+ready:                                                    4024
                in row 5, label cell with "enabled+ready" and apply
                        "module_enabled+ready_indication";

        dec_module_computed:                                                         4026
                in row 5, label cell with "computed" and apply
        "module_computed_indication";
                in row 4, label cell with the computed value and apply
                        "attribute_computed_indication";

        dec_module_value:                                                            4028
                in row 5, label cell with "value" and apply
        "module_value_indication";
                in row 4, label cell with the computed value and apply
"attribute_value_indication";

        dec_module_disabled:                                                         4030
                in row 5, label cell with "disabled" and apply
                "module_disabled_indication";
                in row 4, label cell with "⊥" and apply
"attribute_disabled_indication";

        comp_rule_condition_true:                                                    4032
                to corresponding cell, apply "rule_cond_true_indication";

        comp_rule_contribution_computed:                                             4034
                to corresponding cell, label with computed value and apply
                        "rule_contribution_computed_indication";

        comp_rule_contributed_value:                                                 4036
                to corresponding cell, label with computed value and apply
                        "rule_contributed_value_indication";

        comp_rule_condition_false:                                                   4038
                to corresponding cell, label with "⊥" and apply
"rule_condition_false_indication";

EndCase
```

FIG 40B